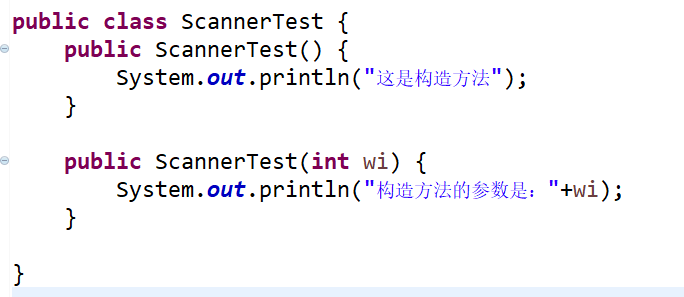
# 类的方法

## 构造方法

1.1 简介

1. 构造方法的方法名必须与类名一样。
2. **构造方法没有返回类型**，也不能定义为void，在方法名前面不声明方法类型。
3. 在创建对象的时候，对象成员可以由构造函数方法进行初始化。
4. 构造方法不能由编程人员调用，而要系统调用。
5. **构造方法可以重载，以参数的个数，类型，或排序顺序区分**。
   1. 单个或多个构造方法



1. 构造方法可以重载。
2. 如果不显式写出构造方法，类默认生成一个没有参数的构造方法。
3. 只要声明构造方法，则类不再默认生成，所以一般都要自己声明一个不带参数的构造方法。

1.3 父类子类的构造方法

(1) 新建子类对象时，会自动调用父类的构造方法，再调用子类的构造方法。

(2) 在子类构造方法中，可以使用关键字super来调用父类的构造方法。

(3) 如果父类构造方法只存在有参数的，则在子类中必须采用super来调用，编译器不会自动调用。

## main方法

1. 对于 public static void main(String[] args)

public: 因为权限必须是最大的。

static: 不需要对象，直接用主函数所属类名调用即可。

void: 主函数没有具体的返回值。

main: 函数名，不是关键字，只是一个 jvm 识别的固定的名字。

String[] arg: 命令行参数。这是主函数的参数列表，是一个数组类型的参数，而且元素都是字符串类型的。

1. main方法是程序的入口，且必须是特定格式才能被JVM识别，只有args可以改动，其他都不能改动。
2. 此外可以重载

public static void main(String[] args){}

public static String main(int[] loli){}

## 方法参数

3.1 形参和实参

(1) 形式参数，用于定义方法的时候使用的参数，目的用来接收调用该方法时传递的参数。

**只有在被调用时才会分配内存单元**，在**调用结束，即刻释放所分配的内存单元**。因此，只在方法内才有效。

(2) 实际参数，用于调用时传递给方法的参数，即传递给被调用方法的值。预先创建并赋予确定值。

* 1. 注意事项

1. 形参变量只有在被调用时才分配内存单元，在调用结束时，即刻释放所分配的内存单元。因此，**形参只在函数内部有效**。函数调用结束返回主调用函数后则不能再使用该形参变量。
2. **实参可以是常量、变量、表达式、函数等**，无论实参是何种类型的量，在进行函数调用时，它们都必须有确定的值，以便把这些值传送给形参。因此**应预先用赋值，输入等办法使参数获得确定值。**
3. **实参和形参在数量上，类型上、顺序上应严格一致**，否则就会发生类型不匹配的错误。
4. 在一般传值调用的机制中只能把实参传送给形参，而不能把形参的值反向地传送给实参。因此在函数调用过程中，形参值发生改变，而实参中的值不会变化。而在引用调用的机制当中是将实参引用的地址传递给了形参，所以任何发生在形参上的改变实际上也发生在实参变量上。

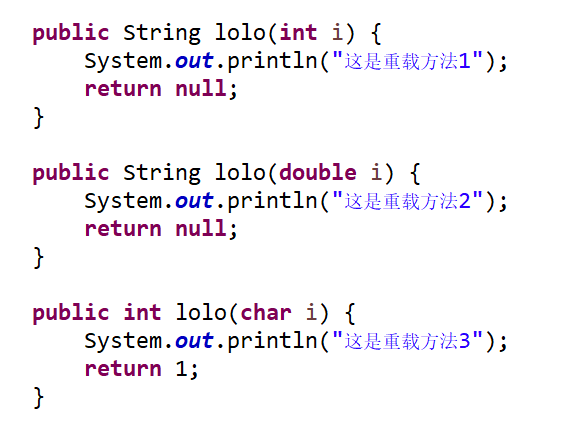
## 方法的重载

方法声明格式：

[修饰符 1 修饰符 2 ....] , 返回值类型 方法名 (形式参数列表) { }

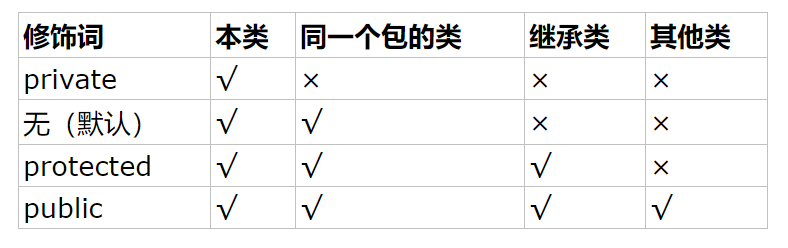
1. 重载的定义

**重载的方法要放在同一个类中，方法名必须相同，参数必须不同，返回值可同可不同。**



# 类的修饰

## 类的访问权限



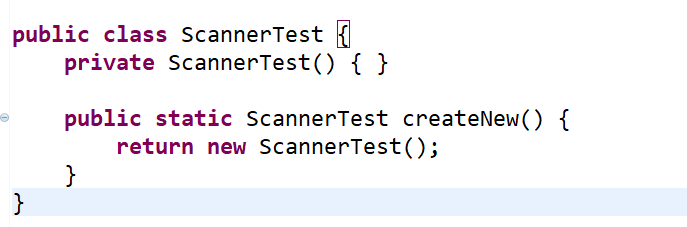
1. Class（类的修饰符）只有三个

可访问控制符是: 公共类修饰符 public

非访问控制符有：抽象类修饰符 abstract //该类是抽象类

最终类修饰符：final //该类不能被继承

1. 对于类的访问权限仅有两个选择：包访问权或是 public。如果不希望其他任何人对该类拥有访问权，**可以把所有的构造器都指定为 private**，从而阻止任何人创建该类的对象。
2. 构造器为私有，在该类的static成员内部仍然可以进行新建对象。



1. 如果一个类的访问权限为 “包访问权”，并且其内部有一个 static 的成员为 public 的话，则其他包中的类仍旧可以访问该 static 成员，哪怕它们并不能生成该类的对象。

## 关键字private

1. 修饰范围：成员变量和成员方法
2. 作用：被修饰的变量或者方法只能在本类中被访问。
3. 主要应用：实体类

## 关键字final

使用范围：用来修饰类，方法，变量

* 1. 修饰类

1. 当用 final 修饰一个类时，表明这个类不能被继承。**final 类中的成员变量可以根据需要设为 final，但是要注意 final 类中的所有成员方法都会被隐式地指定为 final 方法。**
   1. 修饰方法
2. 使用 final 方法的原因有两个。第一个原因是把方法锁定，以防任何继承类修改它的含义；第二个原因是效率。在早期的 Java 实现版本中，会将 final 方法转为内嵌调用。但是如果方法过于庞大，可能看不到内嵌调用带来的任何性能提升。在最近的 Java 版本中，不需要使用 final 方法进行这些优化了。
3. 因此，如果只有在想明确禁止该方法在子类中被覆盖的情况下才将方法设置为 final 的。**类的 private 方法会隐式地被指定为 final 方法。就是不能被重写,但是可以重载。**
   1. 修饰变量
4. 对于一个 final 变量，如果是基本数据类型的变量，则其数值一旦在初始化之后便不能更改；如果是引用类型的变量，则在对其初始化之后便不能再让其指向另一个对象。
   1. final变量和普通变量的区别
5. 当用 final 作用于类的成员变量时，成员变量（注意是类的成员变量，局部变量只需要保证在使用之前被初始化赋值即可）必须在定义时或者构造器中进行初始化赋值，而且 final 变量一旦被初始化赋值之后，就不能再被赋值了
6. 当 final 变量是基本数据类型以及 String 类型时，**如果在编译期间能知道它的确切值，则编译器会把它当做编译期常量使用**。也就是说在用到该 final 变量的地方，相当于直接访问的这个常量，不需要在运行时确定。编译期可以直接访问常量，直接使用。
   1. 被 final 修饰的引用变量

被 final 修饰的引用变量一旦初始化赋值之后就不能再指向其他的对象，但是引用变量指向的那个对象内容可以变，比如实例变量可以改变。

* 1. final 和 static

1. final修饰变量，表示该变量的值是不可改变的，但是可以新建很多个final。
2. static修饰变量，表示该变量在内存中只有一个变量，不能重复的新建，但是可以被修改。